```
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT  LLL
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT  LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRR         TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRRR        TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
```

FILEID**MTHGEXP

```
MM      MM  TTTTTTTTTT  HH      HH    GGGGGGGG  EEEEEEEEEE  XX      XX  PPPPPPPP
MM      MM  TTTTTTTTTT  HH      HH    GGGGGGGG  EEEEEEEEEE  XX      XX  PPPPPPPP
MMMM  MMMM      TT      HH      HH  GG          EE          XX      XX  PP      PP
MMMM  MMMM      TT      HH      HH  GG          EE          XX      XX  PP      PP
MM  MM  MM      TT      HH      HH  GG          EE            XX  XX    PP      PP
MM  MM  MM      TT      HH      HH  GG          EE            XX  XX    PP      PP
MM      MM      TT      HHHHHHHHHH  GG          EEEEEEEE        XX      PPPPPPPP
MM      MM      TT      HHHHHHHHHH  GG          EEEEEEEE        XX      PPPPPPPP
MM      MM      TT      HH      HH  GG  GGGGGG  EE            XX  XX    PP
MM      MM      TT      HH      HH  GG  GGGGGG  EE            XX  XX    PP
MM      MM      TT      HH      HH  GG      GG  EE          XX      XX  PP          ....
MM      MM      TT      HH      HH  GG      GG  EE          XX      XX  PP          ....
MM      MM      TT      HH      HH    GGGGGG    EEEEEEEEEE  XX      XX  PP          ....
MM      MM      TT      HH      HH    GGGGGG    EEEEEEEEEE  XX      XX  PP          ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

(2) 50 HISTORY ; Detailed Current Edit History
(3) 63 DECLARATIONS ; Declarative Part of Module
(4) 233 MTH$GEXP - Standard G Floating EXP
(5) 287 MTH$GEXP_R6 - Special GEXP routine

```
0000     1          .TITLE  MTH$GEXP        ; G Floating Exponential Function
0000     2                                  ; (DEXP)
0000     3          .IDENT /1-006/          ; File: MTHGEXP.MAR     Edit:RNH1006
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;
0000    29 ; FACILITY: MATH LIBRARY
0000    30 ;++
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ;       MTH$GEXP is a function which returns the G floating point
0000    34 ; exponential of its G floating point argument.
0000    35 ; The call is standard call-by-reference.
0000    36 ;
0000    37 ;--
0000    38 ;
0000    39 ; VERSION: 1
0000    40 ;
0000    41 ; HISTORY:
0000    42 ; AUTHOR:
0000    43 ;       Steven B. Lionel, 15-Jan-79: Version 1
0000    44 ;
0000    45 ; MODIFIED BY:
0000    46 ;
0000    47 ;
0000    48 ;
```

```
0000     50          .SBTTL  HISTORY ; Detailed Current Edit History
0000     51
0000     52 ; Edit History for Version 1 of MTH$GEXP
0000     53 ;
0000     54 ; 1-001 - Adapted from MTH$GEXP version 1-008.  SBL 15-Jan-79
0000     55 ; 1-002 - Corrected a typo in the title.  JBS 30-JUL-1979
0000     56 ; 1-003 - Use only through R6.  SBL 21-Sept-1979
0000     57 ; 1-004 - Added large argument logic to avoid lose of significance in
0000     58 ;         EMOD for arguments greater than 2**7.  RNH 24-JUN-81
0000     59 ; 1-005 - Changed W^ to G^ on calls to MTH$$SIGNAL and MTH$$JACKET_TST
0000     60 ;         RNH 09-Sept-1981
0000     61 ; 1-006 - Eliminated symbolic short literals
```

```
                     0000     63          .SBTTL  DECLARATIONS                ; Declarative Part of Module
                     0000     64
                     0000     65 ;
                     0000     66 ; INCLUDE FILES:         MTHJACKET.MAR
                     0000     67 ;
                     0000     68 ; EXTERNAL SYMBOLS:
                     0000     69 ;
                     0000     70          .DSABL  GBL                         ; Declare all EXTRNs explicitly
                     0000     71          .EXTRN  MTH$$SIGNAL                 ; SIGNAL SEVERE error
                     0000     72          .EXTRN  MTH$$JACKET_TST             ; Test to see if called with CALL or
                     0000     73                                              ; JSB
                     0000     74          .EXTRN  MTH$K_FLOUNDMAT             ; Underflow error code
                     0000     75          .EXTRN  MTH$K_FLOOVEMAT             ; Overflow error code
                     0000     76
                     0000     77 ; EQUATED SYMBOLS:
                     0000     78
0000407C             0000     79          ACMASK  = ^M<IV, R2, R3, R4, R5, R6>    ; .ENTRY mask + int ovf enable
00001760             0000     80          X_16LOG2E =     ^0013540            ; Extension for operand in EMODG
                     0000     81 ;
                     0000     82 ; MACROS:
                     0000     83          $SFDEF                              ; define SF$ (stack frame) symbols
                     0000     84 ;
                     0000     85 ; PSECT DECLARATIONS:
                     0000     86
                 00000000     87          .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                     0000     88                                              ; program section for math routines
                     0000     89 ;
                     0000     90 ; OWN STORAGE:  none
                     0000     91 ;
                     0000     92 ; CONSTANTS:
                     0000     93
                     0000     94 ;
                     0000     95 ; Table to be used for scaling. These constants here have been
                     0000     96 ; verified by M. Payne using her program ROOT16 on PDP-10.
                     0000     97 ;
0000 4010            0000     98 TABHI:   .WORD   ^0040020,0                  ; 2**(0/16) = 1.0
0000 0000            0004     99          .WORD   0,0
B558 4010            0008    100          .WORD   ^0040020,^0132530          ; 2**(1/16)
890F 6CF9            000C    101          .WORD   ^0066371,^0104417
72B8 4011            0010    102          .WORD   ^0040021,^0071270          ; 2**(2/16)
517A 3C7D            0014    103          .WORD   ^0036175,^0050572
387A 4012            0018    104          .WORD   ^0040022,^0034172          ; 2**(3/16)
6238 6E75            001C    105          .WORD   ^0067165,^0061070
06FE 4013            0020    106          .WORD   ^0040023,^0003376          ; 2**(4/16)
B715 0A31            0024    107          .WORD   ^0005061,^0133425
DEA6 4013            0028    108          .WORD   ^0040023,^0157246          ; 2**(5/16)
3422 4C12            002C    109          .WORD   ^0046022,^0032042
BFDA 4014            0030    110          .WORD   ^0040024,^0137732          ; 2**(6/16)
2A27 D536            0034    111          .WORD   ^0152466,^0025047
AB07 4015            0038    112          .WORD   ^0040025,^0125407          ; 2**(7/16)
5429 DD48            003C    113          .WORD   ^0156510,^0052051
A09E 4016            0040    114          .WORD   ^0040026,^0120236          ; 2**(8/16)
3BCC 667F            0044    115          .WORD   ^0063177,^0035714
A114 4017            0048    116          .WORD   ^0040027,^0120424          ; 2**(9/16)
0186 73EB            004C    117          .WORD   ^0071753,^0000606
ACE5 4018            0050    118          .WORD   ^0040030,^0126345          ; 2**(10/16)
A0DB 422A            0054    119          .WORD   ^0041052,^0120333
```

```
C491 4019  0058   120          .WORD   ^0040031,^0142221       ; 2**(11/16)
F090 82A3  005C   121          .WORD   ^0101243,^0170220
E89F 401A  0060   122          .WORD   ^0040032,^0164237       ; 2**(12/16)
D3AD 995A  0064   123          .WORD   ^0114532,^0151655
199B 401C  0068   124          .WORD   ^0040034,^0014633       ; 2**(13/16)
529C DD85  006C   125          .WORD   ^0156605,^0051234
5818 401D  0070   126          .WORD   ^0040035,^0054030       ; 2**(14/16)
A487 DCFB  0074   127          .WORD   ^0156373,^0122207
A4AF 401E  0078   128          .WORD   ^0040036,^0122257       ; 2**(15/16)
90D9 A2A4  007C   129          .WORD   ^0121244,^0110331
           0080   130
           0080   131 ;+
           0080   132 ;  \\\ NOTE!!!:  The decimal equivalents in TABLO are taken
           0080   133 ;  directly from MTH$DEXP.  They do not correspond to the
           0080   134 ;  actual value of the G floating equivalent.  When an entry
           0080   135 ;  in TABHI and its corresponding entry in TABLO are added, they
           0080   136 ;  should equal the correct fractional power of 2 to 74 bits. \\\
           0080   137 ;-
           0080   138
           0080   139
0000 0000  0080   140 TABLO:  .WORD   0,0                      ; DECIMAL: 0.D0
0000 0000  0084   141          .WORD   0,0
A62E 3CB8  0088   142          .WORD   ^0036270,^0123056       ; DECIMAL: 0.2252169616881804D-17
0000 0000  008C   143          .WORD   0,0
9BF0 3CCF  0090   144          .WORD   ^0036317,^0115760       ; DECIMAL:-0.2712242510500122D-17
0000 0000  0094   145          .WORD   0,0
B07F 3CB9  0098   146          .WORD   ^0036271,^0130177       ; DECIMAL: 0.5861402647731367D-17
0000 0000  009C   147          .WORD   0,0
F46B 3CA6  00A0   148          .WORD   ^0036246,^0172153       ; DECIMAL: 0.1206457647223494D-16
0000 0000  00A4   149          .WORD   0,0
DA09 3CAA  00A8   150          .WORD   ^0036252,^0155011       ; DECIMAL:-0.8930877995013540D-17
0000 0000  00AC   151          .WORD   0,0
4398 3C9D  00B0   152          .WORD   ^0036235,^0041630       ; DECIMAL:-0.2373071989573779D-17
0000 0000  00B4   153          .WORD   0,0
324C 3CB6  00B8   154          .WORD   ^0036266,^0031114       ; DECIMAL:-0.6257240830881880D-17
0000 0000  00BC   155          .WORD   0,0
1166 3CC6  00C0   156          .WORD   ^0036306,^0010546       ; DECIMAL:-0.1340620676392399D-16
0000 0000  00C4   157          .WORD   0,0
FAA2 3CCE  00C8   158          .WORD   ^0036316,^0175242       ; DECIMAL:-0.7084371812598154D-17
0000 0000  00CC   159          .WORD   0,0
E9F1 3CB6  00D0   160          .WORD   ^0036266,^0164761       ; DECIMAL:-0.3768379065187162D-17
0000 0000  00D4   161          .WORD   0,0
7C47 3C9C  00D8   162          .WORD   ^0036234,^0076107       ; DECIMAL:-0.3048384309613603D-17
0000 0000  00DC   163          .WORD   0,0
A1CD 3CB7  00E0   164          .WORD   ^0036267,^0120715       ; DECIMAL:-0.1276624235300040D-17
0000 0000  00E4   165          .WORD   0,0
1066 3CA1  00E8   166          .WORD   ^0036241,^0010146       ; DECIMAL: 0.1845830375854930D-17
0000 0000  00EC   167          .WORD   0,0
ED03 3CA2  00F0   168          .WORD   ^0036242,^0166403       ; DECIMAL: 0.5075495866202897D-17
0000 0000  00F4   169          .WORD   0,0
B1EE 3CC0  00F8   170          .WORD   ^0036300,^0130756       ; DECIMAL: 0.4822843060675619D-17
0000 0000  00FC   171          .WORD   0,0
           0100   172
           0100   173 ;
           0100   174 ; Constants used in evaluation of polynomials - small arguments
           0100   175 ;
0259 3F1A  0100   176 GXPTB1: .WORD   ^0037432,^0001131
```

```
                                                                                                                
9943 0B6E  0104    177              .WORD   ^0005556,^0114503       ; DECIMAL:    0.2480427857745020D-04
0270 3F4A  0108    178              .WORD   ^0037512,^0001160
32C6 29E6  010C    179              .WORD   ^0024746,^0031306       ; DECIMAL:    0.1984369200268758D-03
C16C 3F76  0110    180              .WORD   ^0037566,^0140554
1C67 143A  0114    181              .WORD   ^0012072,^0016147       ; DECIMAL:    0.1388888879690042D-02
1111 3FA1  0118    182              .WORD   ^0037641,^0010421
DE80 0EA0  011C    183              .WORD   ^0007240,^0157200       ; DECIMAL:    0.8333333262370290D-02
5555 3FC5  0120    184              .WORD   ^0037705,^0052525
5C8F 5555  0124    185              .WORD   ^0052525,^0056217       ; DECIMAL:    0.4166666666667950D-01
5555 3FE5  0128    186              .WORD   ^0037745,^0052525
602C 5555  012C    187              .WORD   ^0052525,^0060054       ; DECIMAL:    0.1666666666667437D-00
0000 4000  0130    188              .WORD   ^0040000,0
0000 0000  0134    189              .WORD   0,0                     ; DECIMAL:    0.5000000000000000D-00
FFFF 400F  0138    190              .WORD   ^0040017,^0177777
FFFF FFFF  013C    191              .WORD   ^0177777,^0177777       ; DECIMAL:    0.1000000000000000D+01
0000 4010  0140    192              .WORD   ^0040020,0
0000 0000  0144    193              .WORD   0,0                     ; DECIMAL:    0.1000000000000000D+01
 00000009  0148    194 GXPLN1=<.-GXPTB1>/8                          ; no. of entries in table
           0148    195
           0148    196 ;
           0148    197 ; Constants used in evaluation of polynomial - regular args
           0148    198 ;
           0148    199
           0148    200
2C4D 3CD6  0148    201 GXPTAB: .WORD   ^0036326,^0026115
F542 E47E  014C    202              .WORD   ^0162176,^0172502       ; DECIMAL:    0.3077130709430240D-15
FD3A 3D4F  0150    203              .WORD   ^0036517,^0176472
F0E2 B8D3  0154    204              .WORD   ^0134323,^0170342       ; DECIMAL:    0.5682419384166091D-13
3091 3DC4  0158    205              .WORD   ^0036704,^0030221
35C8 2F02  015C    206              .WORD   ^0027402,^0032710       ; DECIMAL:    0.9181219559808114D-11
D87F 3E35  0160    207              .WORD   ^0037065,^0154177
F9D7 E6D1  0164    208              .WORD   ^0163321,^0174727       ; DECIMAL:    0.1271587192556359D-08
B2AB 3EA3  0168    209              .WORD   ^0037243,^0131253
4F35 6FBA  016C    210              .WORD   ^0067672,^0047465       ; DECIMAL:    0.1467610032291993D-06
6B08 3F0C  0170    211              .WORD   ^0037414,^0065410
A25A D704  0174    212              .WORD   ^0153404,^0121132       ; DECIMAL:    0.1355080777949815D-04
BFBD 3F6E  0178    213              .WORD   ^0037556,^0137675
C58F FF82  017C    214              .WORD   ^0177602,^0142617       ; DECIMAL:    0.9383847928089872D-03
2E42 3FC6  0180    215              .WORD   ^0037706,^0027102
39EF FEFA  0184    216              .WORD   ^0177372,^0034757       ; DECIMAL:    0.4332169878499658D-01
0000 0000  0188    217              .WORD   0,0
0000 0000  018C    218              .WORD   0,0                     ; DECIMAL:    0
 00000009  0190    219 GXPLN=<.-GXPTAB>/8                           ; no. of entries in table
           0190    220
           0190    221
           0190    222 G_16LOG2_E:                                  ; LOG2(E) * 16
      1547 4057    0190    223              .WORD   ^0040127,^0012507
      82FE 652B    0194    224              .WORD   ^0062453,^0101376
           0198    225
           0198    226 G_LN2_OV_16_HI:
0000FEFA 2E423FC6  0198    227              .QUAD   ^X0000FEFA2E423FC6      ; Hi 39 bits of ln2/16
           01A0    228
           01A0    229 G_LN2_OV_16_LO:
3B3ABC9E F79A3D5C  01A0    230              .QUAD   ^X3B3ABC9EF79A3D5C      ; Low bits of ln2/16
           01A8    231
```

```
                        01A8    233          .SBTTL  MTH$GEXP - Standard G Floating EXP
                        01A8    234
                        01A8    235
                        01A8    236 ;++
                        01A8    237 ; FUNCTIONAL DESCRIPTION:
                        01A8    238 ;
                        01A8    239 ; EXP  - G floating point function
                        01A8    240 ;
                        01A8    241 ; Uses a Chebyshev approximation, with overhang on last step.
                        01A8    242 ;
                        01A8    243 ;
                        01A8    244 ; CALLING SEQUENCE:
                        01A8    245 ;
                        01A8    246 ;       Exponential.wg.v = MTH$GEXP(x.rg.r)
                        01A8    247 ;
                        01A8    248 ; INPUT PARAMETERS:
                        01A8    249
                00000004 01A8   250          LONG = 4                        ; define longword multiplier
                00000004 01A8   251          x = 1 * LONG                    ; Contents of x is the argument
                        01A8    252
                        01A8    253 ; IMPLICIT INPUTS:      none
                        01A8    254 ;
                        01A8    255 ; OUTPUT PARAMETERS:
                        01A8    256 ;
                        01A8    257 ;       VALUE:  G floating exponential of the argument
                        01A8    258 ;
                        01A8    259 ; IMPLICIT OUTPUTS:     none
                        01A8    260 ;
                        01A8    261 ; SIDE EFFECTS:
                        01A8    262 ;
                        01A8    263 ; Signals:  MTH$_FLOOVEMAT if X > 709  with reserved operand in R0/R1 (copied
                        01A8    264 ; to  the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).  Associated
                        01A8    265 ; message is: "FLOATING OVERFLOW IN MATH LIBRARY". Result is reserved operand
                        01A8    266 ; -0.0 unless a user supplied (or any) error handler changes CHF$L_MCH_R0/R1.
                        01A8    267 ;       MTH$_FLOUNDMAT if X =< -709  and caller has hardware enable set.  The
                        01A8    268 ; result  is  set to +0.0. Associated message is: "FLOATING UNDERFLOW IN MATH
                        01A8    269 ; LIBRARY"
                        01A8    270 ;
                        01A8    271 ; NOTE: This procedure disables floating point underflow, enable integer
                        01A8    272 ; overflow, causes no floating overflow or other arithmetic traps, and
                        01A8    273 ; preserves enables across the call.
                        01A8    274 ;
                        01A8    275 ;--
                        01A8    276
                        01A8    277
                    407C 01A8   278          .ENTRY  MTH$GEXP, ACMASK        ; standard call-by-reference entry
                        01AA    279                                          ; disable DV (and FU), enable IV
                        01AA    280          MTH$FLAG_JACKET                 ; flag that this is a jacket procedure
                        01AA    281
    6D  00000000'GF  9E 01AA    282          MOVAB   G^MTH$$JACKET_HND, (FP)
                        01B1    283                                          ; set handler address to jacket
                        01B1    284                                          ; handler
                        01B1    285
                        01B1    286                                          ; in case of an error in special JSB
                        01B1    287                                          ; routine
    50  04 BC 50FD 01  10 01B6 288          MOVG    @x(AP), R0              ; R0/R1 = user's arg
                        01B6    289          BSBB    MTH$GEXP_R6             ; R0/R1 = special EXP(R0/R1)
```

```
04  01B8   285          RET                    ; return - result in R0/R1
```

```
                     01B9   287          .SBTTL  MTH$GEXP_R6 - Special GEXP routine
                     01B9   288
                     01B9   289 ; Special GEXP - used by the standard, and direct interfaces.
                     01B9   290 ;
                     01B9   291 ; CALLING SEQUENCE:
                     01B9   292 ;       save anything needed in R0:R6
                     01B9   293 ;       MOVG    ...,R0                  ; input in R0
                     01B9   294 ;       JSB     MTH$GEXP_R6
                     01B9   295 ;       return with result in R0/R1
                     01B9   296 ;
                     01B9   297 ; Note: This routine is written to avoid causing any integer overflows,
                     01B9   298 ; floating overflows, or floating underflows or divide by 0 conditions,
                     01B9   299 ; whether enabled or not.
                     01B9   300 ;
                     01B9   301 ; REGISTERS USED:
                     01B9   302 ;       R0/R1 - floating argument, then result
                     01B9   303 ;       R2/R3 - temp
                     01B9   304 ;       R5 - integer scratch
                     01B9   305 ;       R6 - integer part of X * LG2(E) * 16   (16I+J)
                     01B9   306 ;
                     01B9   307 ;
                     01B9   308 MTH$GEXP_R6::                           ; special GEXP routine
                     01B9   309 MTH$GEXP_R7::                           ; Release 1 name
                     01B9   310 ;+
                     01B9   311 ;  The preliminary test for overflow works as follows:  First, the sign bit is
                     01B9   312 ;  cleared leaving  the  first word of the |X|.  Then, 1024-4 (bias-4) is sub-
                     01B9   313 ;  tracted, leaving an exponent biased by 4 in bits 14:4 and  the  first  four
                     01B9   314 ;  fraction bits in 3:0. This rebiased value is compared against 230 (decimal).
                     01B9   315 ;  The comparison can have 3 outcomes.   If the rebiased value is now negative,
                     01B9   316 ;  this  means  that the true exponent is < -4 - this is a BLSSU test.  If the
                     01B9   317 ;  rebiased value is positive, but greater than 230 (decimal), then the |X| is
                     01B9   318 ;  greater than or equal to 736, which is  guaranteed  overflow  or  underflow,
                     01B9   319 ;  depending on the sign of X - this is a BLSS test. Otherwise, X is somewhere
                     01B9   320 ;  in the range for the standard evaluation, and flow continues.
                     01B9   321 ;
                     01B9   322 ;-
52   50   8000 8F    AB  01B9   323          BICW3   #^X8000, R0, R2          ; Preliminary test for over/underflow
                     01BF   324                                              ; R2 = exponent bits only
53   52   3FC0 8F    A3  01BF   325          SUBW3   #^X3FC0, R2, R3         ; R3 = 4 + unbiased exponent
     53   00E6 8F    B1  01C5   326          CMPW    #^XE6, R3               ; unsigned compare of |X| with 732
                  6B    1F  01CA   327          BLSSU   SMTST                   ; to more tests if LSSU
                     01CC   328                                              ;  else, -4 < unbiased exp < 11
                     01CC   329                                              ;  no exceptions in EMODG or APPROX
                     01CC   330
          4080 8F    52  B1  01CC   331          CMPW    R2, #^X4080             ; Check for loss of significance in
                     01D1   332                                              ; EMOD ( |X| >= 2**7
                  28    19  01D1   333          BLSS    EVAL                    ; No loss of significance
                     01D3   334
                     01D3   335 ;
                     01D3   336 ; |X| >= 2**4,  EMOD will lose significance so the interger and fractional
                     01D3   337 ; parts of X*16/ln2 must be obtained in seperate steps.
                     01D3   338 ;
52   50     B9 AF  45FD  01D3   339          MULG3   G_16LOG2_E, R0, R2      ; Get integer part of X*16/ln2 in
          56   52 4AFD  01D9   340          CVTGL   R2, R6                  ;    R6 (=I+J) as a longword and in
          52   56 4EFD  01DD   341          CVTLG   R6, R2                  ;    R2/R3 in G format
54   52     B3 AF  45FD  01E1   342          MULG3   G_LN2_OV_16_HI, R2, R4  ; Get fraction part of X*16/ln2 =
          50   54 42FD  01E7   343          SUBG2   R4, R0                  ;    16/ln2*[ X - (I+J)*ln2/16 ]
```

```
                52   B1 AF 44FD  01EB   344          MULG2   G_LN2_OV_16_LO, R2      ;    in R0/R1.
             50 52   42FD  01F0   345          SUBG2   R2, R0                      ;
             50 98 AF 44FD  01F4   346          MULG2   G_16LOG2_E, R0              ;
                   0A   11  01F9   347          BRB     APPROX                      ;
                            01FB   348
50  56  50  1760 8F   91 AF 54FD  01FB   349 EVAL:   EMODG   G_16LOG2_E, #X_16LOG2E, R0, R6, R0
                            0205   350                                             ; get X*16*LG2(E) with
                            0205   351                                             ;  integer part in R6 (=16I+J)
                            0205   352                                             ;  fraction in R0/R1
                            0205   353
        FF3C CF    08   50 55FD  0205   354 APPROX: POLYG   R0,#GXPLN-1,GXPTAB      ; use Chebyshev series
                            020C   355                                             ;  with last coefficient 0
                            020C   356                                             ;  so that last ADDG has overhang
                            020C   357
    55  56   FFFFFFF0 8F    CB  020C   358          BICL3   #-16, R6, R5            ; R5 = J
          50   FDE6 CF45 44FD  0214   359          MULG2   TABHI[R5], R0           ; else MUL by 2**(J/16)
          50   FE5F CF45 40FD  021B   360          ADDG2   TABLO[R5], R0           ; add in LO of 2**(J/16)
          50   FDD8 CF45 40FD  0222   361          ADDG2   TABHI[R5], R0           ; and then HI of 2**(J/16)
                            0229   362
                56   0F   CA  0229   363          BICL    #15, R6                 ; R6 = I
                     08   13  022C   364          BEQL    20$                     ; if I=0, then done
                50   56   C0  022E   365          ADDL2   R6, R0                  ; Add I to exponent.
                            0231   366                                             ; MUL by 2**I by exponent addition
                0F   50   B1  0231   367          CMPW    R0, #^XF                ; test for over/underflow
                     40   15  0234   368          BLEQ    EXCEPT                  ; see what exception is if neg or = 0
                          05  0236   369 20$:     RSB                             ; otherwise return result in R0
                            0237   370
                            0237   371 SMTST:
                     14   19  0237   372          BLSS    20$                     ; exception if exp+4 > 14
              3C80 8F   52   B1  0239   373          CMPW    R2, #^X3C80             ; eliminate underflow from APPROX1
                     08   19  023E   374          BLSS    10$                     ; bypass if E**ARG = 1
                            0240   375
                            0240   376 ;+
                            0240   377 ; Use Chebyshev series for small arg
                            0240   378 ;-
                            0240   379
    FEB9 CF    08   50 55FD  0240   380          POLYG   R0,#GXPLN1-1,GXPTB1     ;Use Chebyshev series
                            0247   381                                             ;  last term is 1; this will
                            0247   382                                             ;  give desired overhang.
                          05  0247   383          RSB                             ; answer is OK, return
                            0248   384
                50   08 50FD  0248   385 10$:     MOVG    S^#1, R0                ; E**X is 1, store it
                          05  024C   386          RSB                             ; and return
                            024D   387
                            024D   388
                            024D   389 ;
                            024D   390 ; Handlers for software detected over/underflow conditions follow
                            024D   391 ;
                50 53FD  024D   392 20$:     TSTG    R0                      ; if big ARG > 0 goto OVERFLOW
                28   18  0250   393          BGEQ    OVER
                            0252   394 ;
                            0252   395 ; Underflow; if user has FU set, signal error.  Always return 0.0
                            0252   396 ;
                            0252   397 UNDER:
                52   DC  0252   398          MOVPSL  R2                      ; R2 = user's or jacket routine's PSL
 00000000'GF    00   FB  0254   399          CALLS   #0, G^MTH$$JACKET_TST   ; R0 = TRUE if JSB from jacket routine
           04   50   E9  025B   400          BLBC    R0, 10$                 ; branch if user did JSB
```

```
                52   04 AD   3C  025E   401          MOVZWL  SF$W_SAVE_PSW(FP), R2   ; get user PSL saved by CALL
                     50   7C  0262   402 10$:       CLRQ    R0                      ; R0 = result, LIB$SIGNAL will save in
                          0264   403                                                  ; CHF$L_MCH_R0/R1 so any handler can fixup
                0D 52   06   E1  0264   404          BBC     #6, R2, 20$             ; has user enabled floating underflow?
                        6E   DD  0268   405          PUSHL   (SP)                    ; yes, return PC from special routine
                7E   00'8F   9A  026A   406          MOVZBL  #MTH$K_FLOUNDMAT, -(SP) ; trap code for hardware floating underflow
                             026E   407                                              ; convert to MTH$_FLOUNDMAT (32-bit VAX-11
                             026E   408                                              ; exception code)
00000000'GF     02   FB  026E   409          CALLS   #2, G^MTH$$SIGNAL               ; signal (condition, PC)
                     05  0275   410 20$:       RSB                             ; return
                         0276   411
                         0276   412 EXCEPT:
                     56  D5  0276   413          TSTL    R6                      ; test sign of I; if I < 0
                     D8  19  0278   414          BLSS    UNDER                   ; go to underflow handler
                         027A   415 ;
                         027A   416 ; Signal floating overflow, return reserved operand, -0.0
                         027A   417 ;
                         027A   418 OVER:                                        ; else process for overflow
                        6E   DD  027A   419          PUSHL   (SP)                    ; return PC from special routine
                  7E   00'8F   9A  027C   420          MOVZBL  #MTH$K_FLOOVEMAT, -(SP) ; hardware floating overflow
              50   01   0F   79  0280   421          ASHQ    #15, #1, R0             ; R0/R1 = result = reserved operand
                         0284   422                                              ; -0.0. R0/0 will be copied to
                         0284   423                                              ; signal mechanism vector (CHF$L_MCH_R0/R1)
                         0284   424                                              ; so can be fixed up by any error
                         0284   425                                              ; handler
00000000'GF     02   FB  0284   426          CALLS   #2, G^MTH$$SIGNAL               ; signal (condition, PC)
                     05  028B   427          RSB                             ; return - R0 restoredd from CHF$L_MCH_R0/R1
                         028C   428
                         028C   429
                         028C   430          .END
```

```
ACMASK          =  0000407C
APPROX             00000205  R      02
EVAL               000001FB  R      02
EXCEPT             00000276  R      02
GXPLN           =  00000009
GXPLN1          =  00000009
GXPTAB             00000148  R      02
GXPTB1             00000100  R      02
G_16LOG2_E         00000190  R      02
G_LN2_OV_16_HI     00000198  R      02
G_LN2_OV_16_LO     000001A0  R      02
LONG            =  00000004
MTH$$JACKET_HND    ********    X    02
MTH$$JACKET_TST    ********    X    00
MTH$$SIGNAL        ********    X    00
MTH$GEXP           000001A8  RG     02
MTH$GEXP_R6        000001B9  RG     02
MTH$GEXP_R7        000001B9  RG     02
MTH$K_FLOOVEMAT    ********    X    00
MTH$K_FLOUNDMAT    ********    X    00
OVER               0000027A  R      02
SF$W_SAVE_PSW   =  00000004
SMTST              00000237  R      02
TABHI              00000000  R      02
TABLO              00000080  R      02
UNDER              00000252  R      02
X               =  00000004
X_16LOG2E       =  00001760
```

+-----------------+
! Psect synopsis !
+-----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_MTH$CODE	0000028C (652.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

+-------------------------+
! Performance indicators !
+-------------------------+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.11	00:00:00.88
Command processing	104	00:00:00.64	00:00:04.12
Pass 1	129	00:00:01.95	00:00:06.65
Symbol table sort	0	00:00:00.03	00:00:00.04
Pass 2	92	00:00:01.15	00:00:02.39
Symbol table output	4	00:00:00.04	00:00:00.09
Psect synopsis output	2	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	364	00:00:03.95	00:00:14.22

The working set limit was 900 pages.
8602 bytes (17 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 56 non-local and 5 local symbols.
490 source lines were read in Pass 1, producing 13 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	4

88 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:MTHGEXP/OBJ=OBJ$:MTHGEXP MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MSRC$:
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MTHGCONJG
LIS

MTHGINT
LIS

MTHGMOD
LIS

MTHEXP
LIS

MTHFLOOR
LIS

MTHDTAN
LIS

MTHDTANH
LIS

MTHGEXP
LIS

MTHGFLOOR
LIS

MTHGMIN1
LIS

MTHGCOSH
LIS

MTHGLOG
LIS

MTHGACOS
LIS

MTHGASIN
LIS

MTHGNINT
LIS

MTHGATAN
LIS

MTHGATANH
LIS

MTHGMAX1
LIS